UNIVERSITÀ DI PISA

Neural color comparator

Intelligent Systems Project

Academic year 2017-2018

Alessandro Noferi, Fabio Condomitti

Contents

1 The project

1.1 Scenario

Let us consider a master color and an industrial process that prints copies of the master. The more accurate the process, the more similar each copy should be to the master. The similarity is mainly evaluated by means of visual checks, where copies are compared to the master, by qualified operators.

1.2 Measure difference of two colors

The *CIE L*a*b*[1] color space let us to describe a color by means of a triple of coordinates on a sphere. By calculating the Euclidean distance is possible to measure the difference between two colors.
Let (L_1, a_1, b_1) and (L_2, a_2, b_2) be two colors in the *CIE L*a*b** color space, then their difference, the ΔE_{ab}, is:

$$\Delta E_{ab} = \sqrt{(L_2^* - L_1^*)^2 + (a_2^* - a_1^*)^2 + (b_2^* - b_1^*)^2} \tag{1}$$

From a perceptual point of view, it holds that:

- when $0 < \Delta E_{ab} < 1$, an observer does not notice the difference;
- when $1 \leq \Delta E_{ab} < 2$, only experienced observers can notice the difference;
- when $2 \leq \Delta E_{ab} < 3.5$, unexperienced observers also notice the difference;
- when $3.5 \leq \Delta E_{ab} < 5$, clear difference in color is noticed;
- when $\Delta E_{ab} \geq 5$, an observer notice two different color.

1.3 The given issue

Unfortunately, the rules discussed in the Section 1.2 do not hold everywhere in the *CIE L*a*b* color space. For example, there are regions where an observer perceives a difference between two colors only when they have a value of ΔE_{ab} higher than 3.5. It is the case, e.g., of the dark region of the *CIE L*a*b** color space. Further imprecisions of ΔE_{ab} occur in the blue-violet region and in the yellow one.

[1] https://sensing.konicaminolta.us/blog/identifying-color-differences-using-l-a-b-or-l-c-h-coordinates/

Therefore checks between master and copy could be affected by imprecision and subjectivity, and it is for this reason that the neural network intervenes, designed to improve the evaluation of the ΔE_{ab}.

2 Part I

The first part of this project aims to develop a Neural Network that is able to estimate the difference (Eq. 1) of a master and an altered copy with varying degrees of deviation.

2.1 Initial dataset

The initial dataset is made up of 1269 reflectance spectra. Each spectrum is related to a master color and is measured with 1 nm step, ranging from 380 nm to 800 nm. This results into 421 samples for each spectrum. It consists of a `spectra` matrix ([421 x 1269]), where each column corresponds to a color patch, and each row to one of the 421 samples of the reflectance curve of that color patch.
First of all these colors have been analyzed in order to identify the most representative and variegated samples, so as to be able to train the neural network with a smaller number than 1269 colors. A subset of only 200 colors has been chosen, leading to a more efficient development from the computational point of view. This choice allows us to reduce the amount of data to be managed in order to make faster executions, but with good results.
As shown in Figure 1, these colors have been selected according to a visual inspection of the whole dataset, choosing the ones considered more representative of the entire color space. This was obtained by applying an exhaustive reduction on the darkest and lightest colors, since they do not add relevant information due of their redundancy.

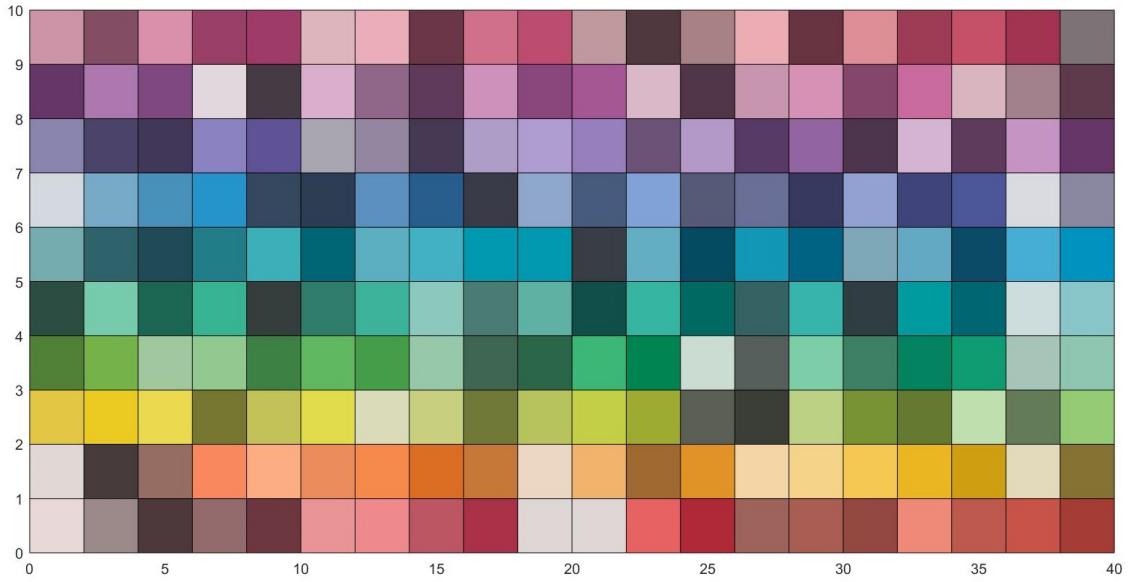


Figure 1: 200 master colors

Later, once the neural network was correctly set up, the cardinality of this subset was increased to manage the complete 1269-color dataset.

2.2 Copies generation

It has been studied how to modify a master color: one way to generate "synthetic" copies of each one consists in adding noise to its spectral reflectance curve.
In order to achieve this result different solutions have been tried: initially the addition of noise was an Additive White Gaussian type (AWGN), point by point, but the network was not able to learn easily due of the high randomness.
The solution was therefore to introduce n intervals, with n to be varied and tested, each one perturbed by a random value. Lower n did not introduce enough aleatory noise distributed among all the wavelengths, while larger n led to lower performance.
The final choice was obtained through the `addNoise` function, which extracts for each color a random value belonging to a uniform distribution, which then, through a Gaussian, is varied for each interval. The function takes as input: a master color, the number of ranges on which we want to operate the perturbation, and the percentage to be added or removed. All of these parameters are relevant for the final results and the computational efficiency to create a balanced training set. It returns the spectra with the noise.

An example of the final result is shown in Figure 2.

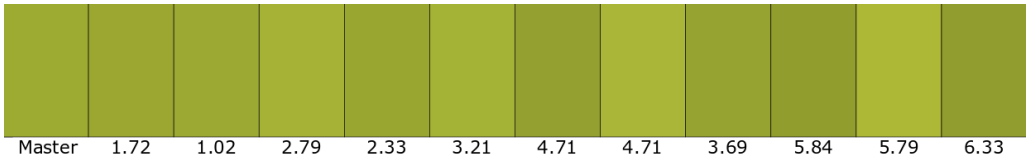


Figure 2: Master color and 11 copies with different ΔE

A conversion from the spectra to RGB has been computed in order to visually compare a master and a copy and see whether the chosen noise addition was enough to change the color. This operations were obtained using the Matlab toolbox `optprop`, which has the function `roo2rgb` that operates this conversion.
Figure 3 shows the differences between master and a copy obtained by varying the noise percentage.

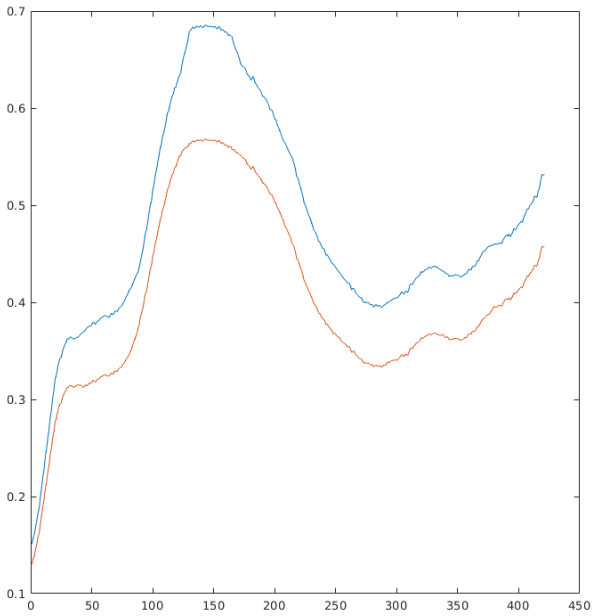


Figure 3: A master and a copy

Since every spectra is composed by 421 variables, a very large data set is going to be obtained, which leads to a high processing time. In order to reduce the degree of complexity a *features extraction* has been performed.

2.3 Features extraction

Feature extraction is the process of collecting relevant and non-redundant information from a set of samples, facilitating the subsequent learning. This operation

obtains a dimensionality reduction, while keeping an accurate and complete description of the original dataset. Statistical methods have been used to achieve this aim. In particular the chosen ones are:

- min
- max
- standard deviation
- mean
- variance
- median

The feature extraction has been applied to each range of the spectrum wavelengths, obtained by dividing the entire visible spectrum in k intervals. Good results have been obtained with $k = 10$. A lower k did not allow to reach good performance, while a higher value has lead to have enough features to describe each color, having better results.
At the end of the process the spectrum size was reduced from 421 to 60 features. The `feature_extraction` Matlab function allows to select the number of ranges in which the spectrum is divided: the most representative results for some extracted features are shown in Figure 4:

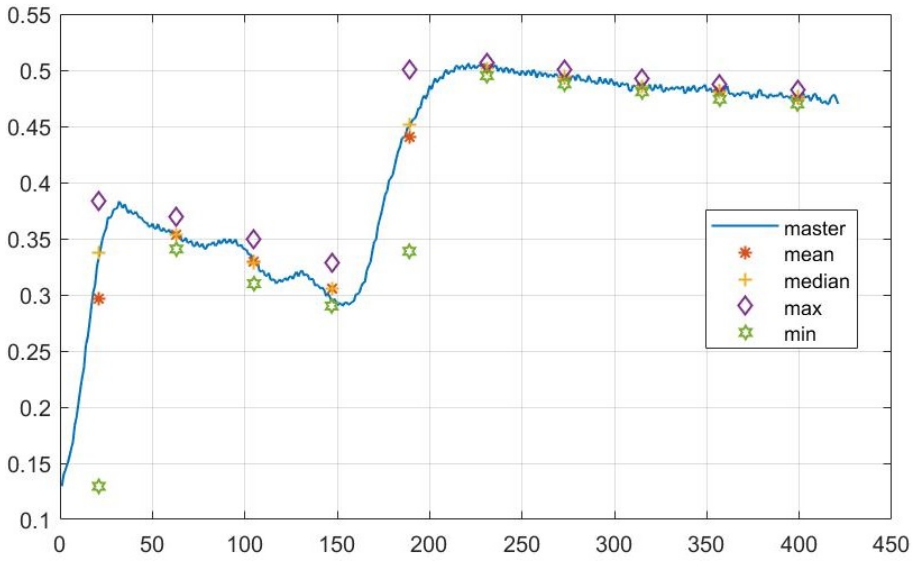


Figure 4: Features extraction

2.4 Redefined dataset

The dataset used for the subsequent feature selection is composed by 12 copies, with different ΔE, for each master color. This is done to obtain a *balanced* data set, with an equal amount of copies for each ΔE range. This constraint lets the network to learn better.

The `master2copy` function both creates the copies for each master and, at the same time, extracts the relative features. The input parameter `dE_r` allows to choose the range on which the ΔE has to vary.

```
[c,c_lab,dE,c_ext,c_rgb]=master2copy(master,noise_r,noise,ext_r,dE_r)
```

2.5 Normalization

Before using the new compact and non-redundant dataset, the inputs of the Neural Network are preprocessed by means of normalization, so that it can efficiently learn. The standardization method used is the *Z-score*, which calculates, for each criterion, the actual value minus the mean of the criterion (centering), divided by its standard deviation (scaling).

2.6 Features selection

Once at this point, `sequentialfs` from Matlab extracts the best features that approximate Formula 1. This function selects a subset of features from the given data matrix that best predict the expected data by sequentially selecting features until there is no improvement in prediction. After some tests, a set of 8 features has been selected. With less features the selected columns were not balanced among the master and each copy. While using 8 features this problem did not occur and the results were good enough.

The `sequentialfs` has been performed 8 times, showing that the selected columns at each step were more or less the same. This approach allowed us to reduce the time needed to execute a complete instance of the `fitnet`, making easier and faster to perform 10 repetition to compute the CIs and the average results during the final considerations, as showed in Table 1. For each noise perturbation, these pre-selected columns should lead to better results in terms of performance.

2.7 Data fitting

Once the relevant information are selected, they can feed a Shallow Neural Network. The Matlab function used is `fitnet(hiddenSize, trainFcn)`, which returns a function fitting neural network with a hidden layer size of `hiddenSize`, tested in section 2.7.1. The training algorithm for the function fitting network is specified with the `trainFcn` parameter. In this case the Levenberg-Marquardt ('`trainlm`') has been used.

2.7.1 Performance evaluation

The performance of the Neural Network are given in terms of *mean squared error* and *regression coefficient*. Each training phase is repeated up to 10 times, since the results do not vary much.

Hidden Layer Size	Regression	MSE	Execution time
3	0.940 ± 0.0021	0.332 ± 0.015	1.852 ± 0.081
4	0.953 ± 0.0013	0.276 ± 0.012	3.600 ± 0.342
6	0.967 ± 0.0036	0.198 ± 0.019	4.402 ± 0.506
8	0.974 ± 0.0005	0.152 ± 0.001	6.037 ± 0.381
10	0.975 ± 0.0004	0.145 ± 0.003	6.217 ± 0.771
12	0.976 ± 0.0009	0.144 ± 0.014	6.785 ± 0.833

Table 1: Neural Network performance with 1269 colors

The results shown in Table 1 are obtained by using the whole dataset of 1269 colors, and they represent the average of 10 repetitions with a 95% confidence interval (CI). The number of hidden neurons, called `hiddenSize`, can be varied in order to exploit a reduction of the computational time required, while still keeping a good *regression coefficient* R. With a large `hiddenSize` the Neural Network will produce more accurate results in terms of R. Low values of `hiddenSize` will visibly reduce the amount of time required to train the network, but on the other side the result accuracy will decrease.

2.7.2 Results

The chosen trade-off between training time and performance is obtained with 8 neurons. By considering a 95% CI, the results are the following (res. 2 and 3):

$$R = 0.974 \pm 0.0005 \tag{2}$$

$$mse = 0.152 \pm 0.001 \tag{3}$$

Since the results of the *training* and *testing* phase do not differ too much, the Neural Network it is not in an *over-training* situation.
One plot of the regression coefficient is shown in Figure 5:

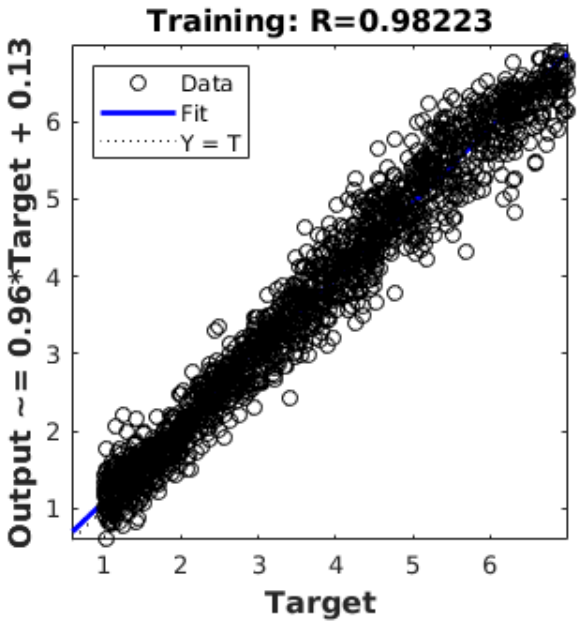


Figure 5: Regression coefficient

3 Part II

3.1 Scenario

The second part aims to improve the formula found in 1. The L^*C^*h color space, similar to CIELAB, is preferred by some industry professionals because its system correlates well with how the human eye perceives color. It has the same diagram as the $L^*a^*b^*$ color space, but uses cylindrical coordinates instead of rectangular coordinates [2]. This is because the CIE $L^*a^*b^*$ color space has some imprecisions in some zone of its sphere, that lead an external observer to experience some inaccuracies[3].

[2]https://sensing.konicaminolta.us/blog/understanding-the-cie-lch-color-space/

[3]https://opentextbc.ca/graphicdesign/chapter/4-4-lab-colour-space-and-delta-e-measurements/

3.2 ΔE inaccuracies

Dark area

In the zone showed in Figure 10a, two colors very similar to a black one lead to experiment, using Formula 1, a $\Delta E \approx 5$. This value does not reflect the real experience of an external observer: they should have a $\Delta E \approx 1$, so the previous formula has to be adjusted keeping in mind this inaccuracies on the dark colors.

Yellow area

Another zone that leads to have some inaccuracies is the yellow area. Looking at the Figure 10b, the Formula 1 returns a $\Delta E \approx 5$, but the perceived difference, in practice, should be smaller than it.

Violet area

In the blue/purple area the opposite behavior occurs: an external observer notices a difference more evident that the result of the Neural Network, which returns a $\Delta E \approx 2.5$, so it should to be higher (Figure 11a).

Unsaturated colors

Color saturation refers to the intensity of color. The unsaturated light colors experience, as the violet area, a ΔE larger than the difference perceived by an external observer, as shown in Figure 11b. As the previous area, it should be higher.

3.3 Fuzzy system

3.3.1 Fuzzy sets

This part aims to improve the Neural Network following the inaccuracies found in the Subsection 3.2. The *fuzzy sets* used are:

- **Luminosity**: it belongs to $[0 : 100]$, following CIE definitions (Figure 6)

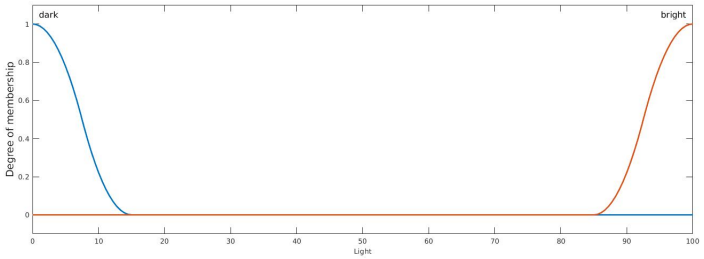


Figure 6: Luminosity membership function

- **Chroma**: its value varies in the range $[0 : C^*_{max}]$, where $C^*_{max} \approx 127$ at $L^* = 50$. Lower or higher values of L^* lead to lower values of C^*_{max}.
  The relationship that links L^* and C^* can be modeled as an ellipse:

$$\frac{x^2}{128^2} + \frac{y^2}{128^2} + \frac{(l-50)^2}{50^2} = 1 \tag{4}$$

  For a given L^*, the relative C_{max} can be obtained as:

$$x^2 + y^2 = 128^2 * (1 - \frac{(l-50)^2}{50^2}) = C^2_{max}$$

  Then, the *Chroma* value has to be expressed as a percentage of the obtained C_{max}: $C^*_{\%} = \frac{C}{C_{max}}$, so it belongs to $[0 : 100]$ (Figure 7).

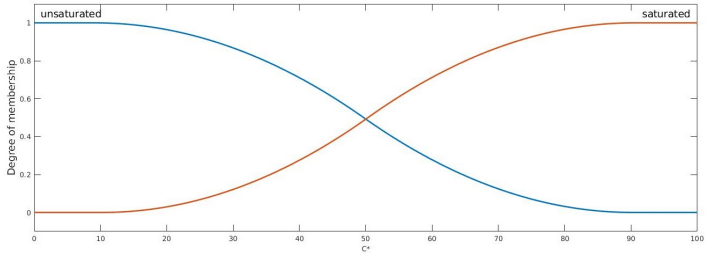


Figure 7: Chroma membership function

- **Hue**: it belongs to $[0° : 360°]$, following CIE definitions (Figure 8)

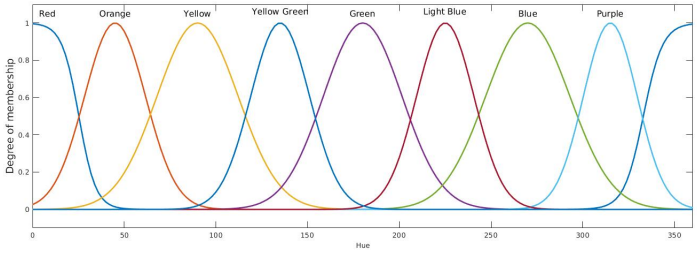


Figure 8: Hue membership function

The *linguistic variables*, that are variables whose values are the fuzzy sets defined above, model the L, C and H. Another one is used for the ΔE given both in input and output to the Fuzzy System, and whose output is also a *linguistic variable* (Figure 9).

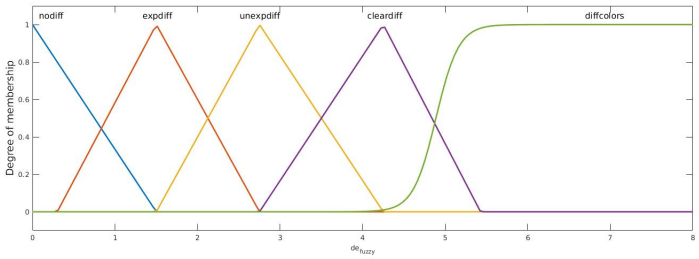


Figure 9: ΔE membership function for both input/output

3.3.2 Fuzzy rules

The rules implemented to improve the Neural Network results by correcting the inaccuracies are showed in Table 2.

L	**C**	**H**	$\mathbf{\Delta E_{math}}$	$\mathbf{\Delta E_{fuzzy}}$
dark	-	-	-	nodiff
not dark	unsaturated	-	expdiff	unexpdiff
not dark	unsaturated	-	unexpdiff	cleardiff
not dark	saturated	yellow	unexpdiff	expdiff
not dark	saturated	yellow	cleardiff	unexpdiff
not dark	saturated	yellow	diffcolors	cleardiff
not dark	saturated	blue	expdiff	unexpdiff
not dark	saturated	blue	unexpdiff	cleardiff
not dark	saturated	purple	expdiff	unexpdiff
not dark	saturated	purple	unexpdiff	cleardiff

Table 2: Rules for the Fuzzy System

3.3.2.1 Selection of correct colors

The rules described in the previous section are very effective in correcting system inaccuracies. On the other side, they may also influence colors that do not require to be adjusted. In order to avoid this useless behavior, a `replaceIndexes` function

has been developed to select among all the dataset the colors that do not need to be adjusted. Among all the 15228 colors (1269×12), composed by masters and copies, 6131 of them are left unchanged by the function, since they were accurate even with the initial system. Then the output of the Fuzzy Inference System is composed with the previous correct results that are left unaltered.

3.4 Final results

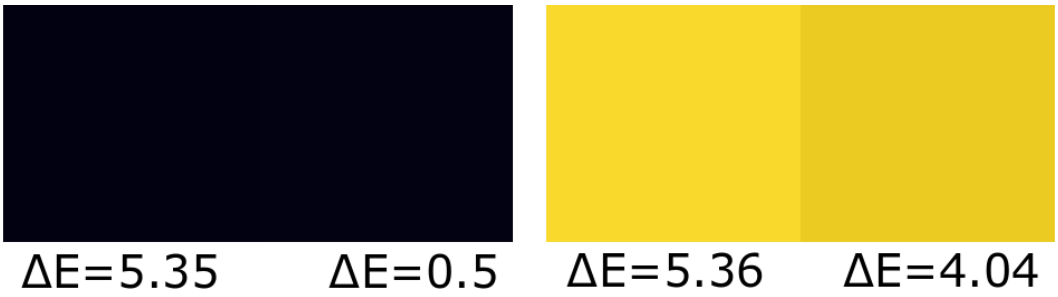
The new dataset is now given as input to the Neural Network described in section 2.7. By considering a 95% CI (10 repetitions), the results are the following (Res. 5 and 6):

$$R = 0.964 \pm 0.0024 \tag{5}$$

$$mse = 0.322 \pm 0.0089 \tag{6}$$

4 Conclusions

It is possible to compare now the results obtained in Part 1 and in Part 2. In the following figures some examples, artificially generated and never used before, are presented:



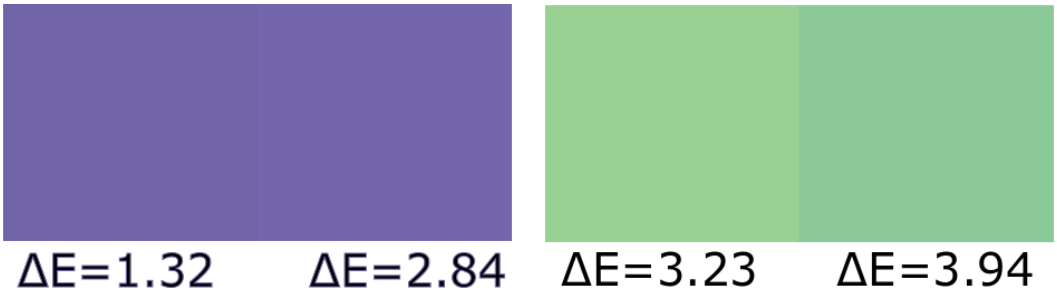
(a) Dark zones (b) Yellow area

Figure 10

In the first zone two color, a dark green and a blue very close to black, have been selected. Before using the Fuzzy System, the $\Delta E \approx 5$, while now clearly shows that the perceived difference it is lower than before. In the example on the right the new ΔE is adjusted towards a lower value, since the two colors experiment a not so large difference.

In the following figures the opposite behavior occurs: these two examples show that the perceived difference should be higher than the one obtained from the Neural

Network used in Section 2.7, so the results have been adjusted towards an higher value.



(a) Violet area (b) Unsaturated colors

Figure 11

At the end the Neural Network behaves as expected, and it is still able to have good performance while correcting the inaccuracies of the previous version.